Towards Privacy Compliant and Anytime Recommender Systems

Armelle Brun, Anne Boyer

Towards Privacy Compliant and Anytime Recommender Systems

Armelle Brun and Anne Boyer

LORIA, Université Nancy2,
615, rue du Jardin Botanique, 54506 Vandoeuvre-lès-Nancy,
{brun,boyer@loria.fr},
WWW home page: http://kiwi.loria.fr

Abstract. Recommendation technologies have traditionally been used in domains such as E-commerce and Web navigation to recommend resources to customers so as to help them to get the pertinent resources. Among the possible approaches is collaborative filtering that does not take into account the content of the resources: only the traces of usage of the resources are considered. State of the art models, such as sequential association-rules and Markov models, that can be used in the frame of privacy concerns, are usually studied in terms of performance, state space complexity and time complexity. Many of them have a large time complexity and require a long time to compute recommendations. However, there are domains of application of the models where recommendations may be required quickly. This paper focuses on the study of how these state of the art models can be adapted so as to be anytime. In that case recommendations can be proposed to the user whatever is the computation time available, the quality of the recommendations increases according to the computation time. We show that such models can be adapted so as to be anytime and we propose several strategies to compute recommendations iteratively. We also show that the computation time needed by these new models is not increased compared to classical ones; even so, it sometimes decreases.

1 Introduction

Web personalization alleviates the burden of information overload by tailoring the information presented to users based on their needs and preferences. In recent years, personalized search has attracted interest in the research community as to provide more effective and efficient information access. Web personalization has applications in various domains such as E-commerce to recommend products to customers or Web navigation to recommend resources to a particular user, named active user.

Recommender systems are a means to perform Web personalization, they generally fall into three categories of content-based systems [1] which make recommendations based on semantic content of data, knowledge-based systems [2] which make recommendations based on knowledge about the active user and pre-established heuristics and collaborative-filtering systems [3] which make recommendations by examining past interactions of the active user with the system, called traces of *usage*.

In this paper, we are especially interested in Web navigation and E-commerce. In that case, collaborative filtering (and usage analysis) seems to be the most appropriate

approach for several reasons. First, resources do not have to be tagged: only the id of each resource has to be known, that is an advantage when the set of resources is large and evolves over time, which is the case on the Web. Second, no *a priori* information has to be known about users (no demographic information, no predefined preferences, etc.). The only information available is the trace of usage of users for the resources. In that case recommender systems perform Web log or usage traces, analysis.

Nowadays, privacy concerns have to be considered by recommender systems [4]. To face this problem, we propose here to store the traces of usage without any information about the corresponding users. More specifically, these traces are stored under the form of anonymous sequences of navigation, privacy is thus respected. The set of sequences of navigation stored is named *training dataset*. Let us notice that there several ways to store navigation sequences so as they are anonymous (on multiple databases for example), the way we choose to store them is the simpler one.

To perform recommendations by using collaborative filtering, data mining techniques are classically used to discover usage patterns that will be exploited to compute predictions. Techniques such as similarity between users or resources [5], Markov models of sequential navigation [6], or sequential association mining [7] have been proposed in the literature.
All these approaches have an off-line part that computes a model/representation of the training dataset that aims at discovering usage (navigation) patterns.
In the on-line part, the algorithm exploits the usage patterns from the off-line part and the information available about the active user is used to compute the interest of each resource for him/her.
In our case, the only information available about the active user is reduced to his/her active session of navigation, as his/her past navigations are stored anonymously in the training dataset.

Among the possible approaches, as the past navigations of each user are stored anonymously, no similarity between users can be computed, thus the approach based on such information cannot be investigated. In the same way, the popular item-based approach that exploits similarities between resources [5] cannot be exploited neither. Indeed, in such an approach the set of resources cannot evolve and votes or ratings on the resources have to be given by users, which is not our case.
However, Markov models and association mining approaches can deal with anonymous navigations and do not require any votes, they can thus be used to compute recommendations.

In the frame of E-commerce or Web navigation, a few amount of time may be available to compute predictions. Indeed, when a user goes on a web page, the recommender has to present him/her the list of recommendations right away, when the page he/she wants to consult appears (the recommendations can then be changed).
Most of recommendation algorithms either run to completion or provide no solution at all. However, computing the optimal recommendation set may require a large amount of computation time and an answer may be required at any moment. Anytime algorithms have the advantage to provide an answer (here a recommendation) whenever the algorithm is stopped and the quality of results improves regularly as computation

time increases. The answer generated by anytime algorithms is an approximation of the optimal answer.

In this article, we are thus interested in anytime privacy-compliant recommendation algorithms in order to match specificities of E-commerce or Web navigation. More specifically, we focus on classical recommendation algorithms that are originally not anytime, and we study how they can be adapted so as to be anytime and thus recommend resources whatever computation time is available.

The second section presents the way recommender systems compute the recommendations. In the third and fourth sections, the two recommendation algorithms we are interested in (sequential association-rules and Markov models) are presented and strategies to make such systems being anytime are put forward. In the fifth section, conclusion and perspectives are detailed.

2 Recommender Systems

The recommendation problem is typically formulated as a prediction task in which a prediction model is built according to the prior training dataset and then the model is used in conjunction with the active user dynamic profile to predict the level of interest (score) of the active user for each resource.

The profile of the active user is made up of the resources he/she has consulted in the active session (as no additional information is known about the user). However, as a session may be long and as the resources that have been consulted early in the session may be less useful than the resources consulted recently, most of the models use a sliding time window that contains the last n resources the active user has consulted. These last n resources approximate his/her dynamic profile. From now on, the dynamic profile of the active user will be called *active profile*.

The recommendation problem can be viewed as follows: given the past consultations of the user, his/her active profile ap, what is/are the most pertinent resource(s) r_m to recommend? The resource(s) to be recommended are the ones with the highest score that maximize the following equation:

$$r_m^* = \underset{r_m}{argmax}\ S(r_m|ap) \tag{1}$$

Where $(S(r_m|ap))$ is the score of the resource r_m given the active profile ap.

The literature is usually interested in the way to estimate $S(r_m|ap)$. In this article, we focus on the way classical approaches can be adapted so as to be anytime. The approaches we are interested in are sequential association-rules and Markov models as they are the most popular data mining approaches and they can be used on anonymous data. Moreover, these algorithms can be used to compute recommendations without requiring the active user is identified.

3 Sequential Association-Rules Based Recommenders

3.1 Sequential Association-Rules

The framework of association rules (AR) was introduced into the data mining community by Agrawal et al [8]. Association rules were used to capture relationships among items based on patterns of co-occurrence across transactions.

Sequential Association-Rules (SAR) are an evolution of AR, in which the order of the items is considered. The framework of SAR mining was introduced by Agrawal et al [7] to capture ordered relationships between items.

A SAR is an expression of the form $X \Rightarrow Y$, where X (called the antecedent) and Y (the consequent) are sequences of resources. Usually $X \Rightarrow Y$ is considered as a SAR if both its support and confidence are above two thresholds that have to be fixed. Support is defined as the number of transactions that contain both X and Y and confidence is the number transactions that contain both X and Y divided by the number of transactions that contain X (confidence can be viewed as the conditional probability of Y given X).

In the frame of usage mining [9], SAR refer to resource associations and capture resource dependencies in sessions of navigation. A SAR means that, when users have consulted the sequence of all resources in X, they usually consult Y. Here, as the goal is to predict the next resource to be seen, Y is made up of a single resource. In SAR, resources can be either contiguous or non-contiguous [10,11,12].

The off-line part of SAR based recommender systems searches all the SAR according to their support and confidence. A large variety of algorithms for mining SAR have been published in the literature, as Apriori [13] and DIS [14] algorithms. These algorithms are incremental and SAR made up of $k+1$ resources are deduced from the SAR of k resources.

3.2 Score Computation

To compute predictions in the on-line part, a score has to be assigned to each resource r_m. SAR based recommender systems exploit the sequence of resources in the active profile of the user, and compare it to the antecedents of the SAR from the off-line part. The rules that match the active profile of the user are then used to compute the score of each resource.
A rule that matches the active profile (called a matching rule) is a rule with an antecedent that is a subsequence of the active profile (ap). Let $SubSeq$ be the function mapping ap to the list of sub-sequences of ap. The resources (r_m) that are consequence of the matching rules are then candidates to be recommended.

To compute the score of each resource $S(r_m|ap)$, the confidence of each matching rule (confidence($X \Rightarrow r_m$)) is used. In the case that several matching rules have the same consequence, several policies are used in the literature:

- The max_policy: given a resource r_m, the recommendation algorithm searches the SAR that have r_m as consequence, with an antecedent X that matches the active profile of the user ($X \in SubSeq(ap)$). Among these SAR, the one with the highest

confidence for resource r_m is retained and the confidence of this SAR is assigned to the score of r_m [15,16]:

$$S(r_m|ap) = \underset{X \in SubSeq(ap)}{max} \text{confidence}(X \Rightarrow r_m) \quad (2)$$

– The sum_policy: as for max_policy, all the SAR with consequent r_m and that have an antecedent that matches the active profile, are retained. The sum of the confidence values over these SAR is computed and then assigned to the score of the resource r_m. This strategy enables to give more weight to the resources that are associated with more rules [17].

$$S(r_m|ap) = \underset{X \in SubSeq(ap)}{sum} \text{confidence}(X \Rightarrow r_m) \quad (3)$$

These works mainly focused on the way to evaluate the "final" score of each resource, by using matching rules. However, they do not focus on the exact way to integrate these rules so as to make the recommender anytime.
Some other works have been interested in anytime recommenders, however they only focus on the off-line part, for example [18] has been interested in anytime SAR mining in peer-to-peer systems.

3.3 Anytime SAR based recommenders

A SAR-based recommender is basically not anytime: the recommender searches all the matching rules $(X \Rightarrow r_m)$ with $X \in SubSeq(ap)$, then it computes the score for each resource r_m according to these matching rules and the policy chosen (max_policy or sum_policy).
Some works have been interested in the way to store rules to improve the access and computation time to construct the recommendation list. For example [19] proposes to store rules in a navigational pattern tree; even so the computation of the recommendation list may take time. To reduce this time, we propose to implement anytime SAR-based recommender systems.
Let us notice that when anytime algorithms run to completion, recommendations are similar to the ones computed by classical approaches.

We propose here three strategies for one anytime rule-based recommender.

Small Antecedent First Strategy (SAF)

The SAF strategy is based on the following hypothesis:
Hypothesis 1: The probability that a resource is the consequence of a SAR with an antecedent made up of only few resources is higher than the probability that a resource is the consequence of a SAR that has an antecedent with a larger number of resources. In other words, given a resource r_m and X_1, X_2 two sequences of resources (with cardinals $|X_1|$ and $|X_2|$) with $X_1 \in SubSeq(ap)$, $X_2 \in SubSeq(ap)$ and $|X_1| \ll |X_2|$. Then, the probability that the rule $(X_1 \Rightarrow r_m)$ exists is greater than the probability that the rule $(X_2 \Rightarrow r_m)$ exists.

An anytime algorithm has to assign a score to each resource early in the process and then refines these scores if time is available. The SAF strategy we propose first assigns to each resource its confidence *a priori*. Let us recall that the confidence of a SAR can be viewed as the conditional probability of the consequence given the antecedent. Thus, in the case the antecedent is empty, the confidence is the probability *a priori* of the consequence. If the algorithm has to stop at this step, recommendations can be presented as a score is assigned to each resource, the probability *a priori* of a resource is a first approximation of the "final" score of each resource.
Second, based on the hypothesis 1, the algorithm uses SAR in the list of matching rules with antecedents of size $k = 1$ so as to refine the score of each resource. The probability of finding a corresponding SAR is still relatively high and the score of each resource is a better approximation of the "final" score than at the end of the preceding step.
The algorithm then iterates on the value of k, by increasing k. At each step k, SAR with an antecedent of size k are used and scores are updated.

Based once more on hypothesis 1, we can notice that the probability that a SAR matches the active profile decreases as the value of k increases. Thus, if there is no time left to compute iterations with high values of k, the score of each resource may be even so a good approximation of the "final" score.

Whenever the algorithm stops, a score is assigned to each resource, and recommendations can be performed, by using Equation (1).

With the SAF strategy, the SAR have to be accessed both via the consequence-resource and the size of the antecedent. We thus propose to store the whole list of SAR in several lists of SAR: one list for each consequence-resource and for each size of antecedent, as presented in Figure 1.

List name	Antec. size	Conseq.-Resource	SAR list
$L_{1,1}$	1	R_1	$((R_x,S(R_x=>R_1))$, $(R_y,S(R_y=>R_1))$ ...)
$L_{1,2}$	2	R_1	$((R_x,R_y,S((R_x,R_y)=>R_1))$, $(R_y,R_z,S((R_y,R_z)=>R_1))$ ...)
$L_{1,3}$	3	R_1	$((R_x,R_y,R_t,S((R_x,R_y,R_t)=>R_1))$, $(R_y,R_z,R_m,S((R_y,R_z,R_m)=>R_1))$ ...)
⋮			
$L_{2,1}$	1	R_2	$((R_x,S(R_x=>R_2))$, $(R_y,S(R_y=>R_2))$ ...)
$L_{2,2}$	2	R_2	$((R_x,R_y,S((R_x,R_y)=>R_2))$, $(R_y,R_z,S((R_y,R_z)=>R_2))$ ...)
$L_{2,3}$	3	R_2	$((R_x,R_y,R_t,S((R_x,R_y,R_t)=>R_2))$, $(R_y,R_z,R_m,S((R_y,R_z,R_m)=>R_2))$ ...)
⋮			

Fig. 1. Lists of SAR stored so as to be managed by the SAF strategy

During the score estimation, given the consequence-resource r_m and the size of the antecedent k, the system traverses the corresponding SAR list $L_{r_m,k}$ to find the SAR that match the active profile.

In the literature, some works store the set of rules in a tree that requires less storage space. Such trees are not optimal for the SAF strategy. Indeed, the access has to be made on the consequence of the rules. We can thus imagine to store rules so as the consequence of the rules are stored at depth 1 in the tree, the last element of the antecedents is stored at depth 2, etc. However, as the algorithms have to perform breadth-first traversal, a classical tree cannot be used as traversal will take time.

The advantage of the SAF strategy is that a score is assigned to each resource early in the process (hypothesis 1). However, this score may not be a reliable approximation of the "final" score, whatever is the policy used (max_policy or sum_policy).

What about the computation time of such an anytime recommender? Usual rule-based recommenders systematically traverse once the whole set of SAR. Concerning the anytime SAF version, if a pointer is placed on each list, that memorize the last SAR used; then if time is available, this algorithm also traverses the whole set of rules (all the lists) once, thus computation time is similar to the classical one.

Highest Confidence First Strategy (HCF)

The HCF strategy is based on the following hypothesis:

Hypothesis 2: The SAR with the highest confidence for a given consequence-resource is a good approximation of the "final" score of a given resource.

With the HCF strategy, the SAR with the highest confidence have to be accessed first to have a good approximation of the final score as early as possible.

SAR have thus to be accessed according to the consequence-resource and their confidence value, the storage has to be different from the one of the SAF strategy (Figure 1). We propose to create one SAR list for each consequence-resource, the elements of a list being ordered according to their confidence value, whatever is the size of the antecedent, as presented in Figure 2.

List name	Conseq.-Resource	SAR lists ordered by confidence values
L_1	R_1	$((R_y,R_z,S((R_y,R_z)=>R_1)), (R_x,S(R_x=>R_1)),$ $(R_y,R_z,R_m,S((R_y,R_z,R_m)=>R_1)), (R_y,S(R_y=>R_1))$ ...)
L_2	R_2	$((R_x,R_y,S((R_x,R_y)=>R_2)), (R_x,R_y,R_z,S((R_x,R_y,R_z)=>R_2)),$ $(R_x,S(R_x=>R_2))$ ...)
L_3	R_3	$((R_t,S(R_t=>R_3)), (R_y,R_z,R_m,S((R_y,R_z,R_m)=>R_3)),$ $(R_y,R_z,R_m,R_t,R_x,S((R_y,R_z,R_m,R_t,R_x)=>R_3))$ ...)
⋮	⋮	⋮

Fig. 2. Lists of SAR stored so as to be managed by the HCF strategy

When the max_policy is chosen, the algorithm considers one list after the other (each list corresponds to one resource). When a SAR in a given list matches the active profile, the algorithm stops the evaluation of the score of the resource and evaluates the

score of another resource. In that case, the algorithm cannot be considered as anytime as the "final" score of each resource is directly found. Indeed, when a matching SAR is found, it is the one that maximizes the confidence, thus it is the one that will be retained by the policy.
However, in the case the sum_policy is chosen, in the first step the algorithm searches the first SAR that matches the active profile, for each resource (as for the max_policy). Then, the score of each resource can be refined step by step while traversing the rest of the lists.
This strategy has the advantage of having a good estimation of the "final" score once a matching rule is found. However, as the highest confidence usually corresponds to a SAR with a large antecedent (see Hypothesis 3 below), and as the probability the active profile matches a SAR with a large antecedent is low (Hypothesis 1), then the computation time required to assign a score to a resource may be large.

In the case the max_policy is used, the algorithm traverses at worst all the lists once (if pointers on lists are stored), thus computation time is similar. However, in many cases a matching rule will be found before the list is completely traversed, thus computation time is lower.
In the case of the sum_policy, the lists are at worst entirely traversed once. Computation time is thus similar to classical ways to compute scores.

Small Antecedent and Highest Confidence First Strategy (SAHCF)

The SAHCF strategy is based on an additional hypothesis:

Hypothesis 3: The highest confidence of a resource r_m is usually provided by a SAR with a large antecedent.

This hypothesis 3 is mixed with the two previous strategies: the SAHCF strategy tends to assign a score to a resource as early as possible in the process (SAF strategy) while using SAR with the highest confidence first.
One list is thus constructed for each resource and for each size of antecedent (as in Figure 1) and the elements of each list are ordered according to their confidence value (as in Figure 2).

If the max_policy is chosen: in the first step the confidence value of the first matching SAR with an antecedent of size 1 (lists L_{*1} in Figure 1) is assigned to the score of the resource. These lists are no more considered when a matching SAR is found. In the second step, the lists L_{*2} are traversed and the step ends when one SAR matches, etc. The hypothesis 3 is not used with the max_policy as the traversal of a list is stopped when a matching SAR is found.

When the sum_policy is chosen, the hypothesis 3 is used: at the end of the first step, a matching SAR is found for each resource (on lists L_{*1}). The question is then: does the second step focus on the rest of lists L_{*1} or does it first search a matching rule in lists L_{*2} ? Following the hypothesis 3, matching SAR are first searched in lists L_{*2}, L_{*3}, etc. then, traversal of lists L_{*1} are ended.

Compared to the SAF strategy, SAHCF runs more quickly in the case of max_policy as the maximal confidence is found early in the process for each list with antecedents

of size k. In the case of sum_policy, the process has similar computation time (still in the case of the use of pointers) and has a better estimation of the "final" score earlier in the process as high confidence values are integrated first in the score.

4 Markov Models

We focus now on the well-known Markov-based recommender systems and study how they can be adapted to provide recommendations whatever is the time allowed to the recommender.

4.1 k order Markov Models

A k-order Markov model (KMM) assumes that the consultation of a resource by the active user is influenced by only and exactly the last k resources he/she has accessed, the resources he/she has consulted before these k resources are considered as non-informative.
A Markov Model is a set of conditional probabilities $P(r_m|X)$, where X is a sequence of resources and X will be called the antecedent, as in the SAR framework. A KMM is thus a set of conditional probabilities where the size of the antecedents is exactly k. These probabilities are learnt during the off-line part, on the training dataset.

On the on-line part, a KMM-based recommender computes the score of a resource to be consulted by the active user given the sequence of his/her exactly k previously accessed resources. The active profile of the user is in that case approximated by the sequence of these k resources. The score of a resource is in that case its conditional probability. Given the set of conditional probabilities, the probability of a resource r_m is the one among the conditional probabilities that has an antecedent equal to the active profile of the user.

The resources that are recommended are the ones that have the highest probability. Obviously, the higher the value of k is, the most accurate the probabilities are (in the case of no sparse data problem), and it has been shown [20] that, when applied to recommender systems, the accuracy of KMM increases with the value of k.

However, the higher the value of k is, the larger the number of states to be stored is and the lower the coverage is (*cf* hypothesis 1: the probability that the active profile of size k perfectly matches one antecedent in the model is low in the case of high values of k).

Markov models are a popular approach used in recommenders, due to their accuracy. A recommender based on Markov models of order k cannot be adapted so as to be anytime as the probability of a resource is directly known: either the sequence of the k last resources the active user has consulted matches the antecedent of one of the conditional probabilities, or it does not. In the case it matches, the probability of each resource is known right away.

4.2 All kth order Markov Models

To cope with the coverage problem of KMM, All-kth-order Markov Models (AKMM) have been proposed in [6]. In AKMM, various KMM of different order k are trained and used to make predictions.

The AKMM are based on the following hypothesis:

Hypothesis 4: The larger the sequence of navigation used to compute prediction is, the more accurate the probability is.

It has been shown that this hypothesis is true in the case of KMM (see section 4.1.

Following hypothesis 4, predictions are thus first computed by using a k-order MM. This step leads to an accurate recommendation as predictions are made by using a large sequence of navigation. If no prediction can be performed (as the coverage in that case is low), a $k-1$ order KMM is used, etc. The value of k is iteratively decreased until a recommendation can be made. The coverage of AKMM is thus highly increased. In AKMM, all sequences of navigation (with their corresponding conditional probability) are stored, whatever is their size (in the limit of k).

The drawback of AKMM is their state space complexity: the number of states dramatically increases with the value of k. However, some works have been interested in the way to store this model to decrease the space complexity. [21] for example stores navigational patterns in a suffix tree and at each node, the conditional probability of this node given its parent node is stored.

AKMM also suffer from time complexity: predictions are first computed with a KMM of order k (hypothesis 4). Due to the coverage problem of high values of k (hypothesis 1), the order of the model is frequently decreased, and of course it takes time. Consequently, if a small time is available to compute predictions, in some cases, no recommendation can be proposed to the user.

In the frame of the storage of navigation patterns in a tree, [22] proposes a real-time recommender algorithm: all navigation patterns are stored in the tree (the tree evolves as new user navigations arrive), confidence and support are computed when traversing the tree to find candidate patterns to compute predictions, that takes time. Moreover, the "final" score of a resource is computed before computing the one of the following resource, the recommendation is thus not anytime.

4.3 Anytime All kth order Markov Model

We propose to adapt the AKMM so as it is anytime. This strategy is based on a hypothesis 1: the probability that the active profile of size k perfectly matches one antecedent in the model is high in the case of low values of k)

Given this hypothesis, we thus propose to inverse the way to use k (the order of the models) by increasing its value instead of decreasing it step by step. Recommendations are first computed with a low order KMM (a 0-order KMM for example, that corresponds to the probability *a priori* of the resources). The use of such a model guarantees a high coverage value. Then if time is available, predictions are refined by using higher-order KMM. At an iteration value k, the recommender searches, in the set of conditional probabilities, the antecedents of size k that match the active profile of size k. If a match is found, the recommender replaces the score of each resource (computed

during iteration step $k-1$) by the new scores (from iteration step k).
Let us notice that at a step value k. The replacement of the score values between two steps guarantees that recommendations will be similar to the ones made with classical AKMM.

Such an approach makes the recommendation process anytime: in the first iteration step, an *a priori* probability is assigned to each resource, recommendations can be made at this step. Then these probabilities are refined if time is available.

Let us notice that the HCF strategy proposed in section 3.3 cannot be applied on AKMM. Indeed, with AKMM the iterations have to be done on the size of the antecedent, thus matching cannot be dependent on the conditional probability.

The computation time of this anytime model is similar to the one of the classical AKMM. Furthermore, the time required to run to completion may be lower than for classical AKMM. In the case of a high value of k, the probability that the active profile matches an antecedent of size k is low, then classical AKMM may have to iterate many times until recommendations can be performed (with a low value of k).
At the opposite, the anytime recommender may run more quickly: at a step k, if no antecedent matches the active profile, the recommender can stop. Indeed, if the active profile of size k does not match any antecedent, thus no antecedent of size greater than k can match the active profile of size greater than k. Thus, the highest value k with a matching antecedent can be reached in a lower number of iterations than with classical AKMM.

5 Conclusion and Perspectives

Data mining approaches are classically used in recommender systems based on collaborative filtering. This article studies the way recommender systems, that use data mining techniques, can be adapted to be anytime. To this end, the frame of this article is first presented: Web recommender systems that deal with privacy concerns. After having presented classical recommender systems, we present why they cannot guarantee to provide recommendations in the case few computation time is available.

We propose several strategies, based on various hypothesis, to compute iteratively recommendations, thus they can be presented to the user, whatever is the time allowed to the recommender. We show that these strategies and these anytime recommenders, have a computation time similar to the one of classical approaches and the computation time until completion is even decreased in some cases.

As a future work, we intend to evaluate the evolution of the quality of recommendations according to the number of iterations, and the strategy used. In addition, we will study how such recommenders can also be incremental and take advantage of the recommendations made at the preceding step to compute recommendations to make in one step.

References

1. Pazzani, M., Billsus, D.: Content-Based Recommendation Systems. In: The Adaptive Web. Springer Berlin / Heidelberg (2007) 325–341

2. Burke, R., Hammond, K., Cooper, E.: Knowledge-based navigation of complex information spaces. In: Proceedings of the 13th National Conference on Artificial Intelligence, Menlo Park, Canada (1996) 462–468
3. Goldberg, D., Nichols, D., Oki, B., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM **35**(12) (1992) 61–70
4. Aïmeur, E., Brassard, G., Fernandez, J., Mani, A.: Alambic : a privacy-preserving recommender system for electronic commerce. International Journal of Information Security **7**(5) (October 2008) 307–334
5. Sarwar, B.M., Karypis, G., Konstan, J.A., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: World Wide Web. (2001) 285–295
6. Pitkow, J., Pirolli, P.: Mining longest repeating subsequences to predict world wide web surfing. In: USITS'99: Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems. (1999)
7. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Proceedings of the 5th International Conference on Extending Database Technology. (1996) 3–17
8. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD Conference on Management od Data. (1993) 207–216
9. Fu, X., Budzik, J., Hammond, K.: Mining navigation history for recommendation. In: Proceedings of the 5th International Conference on Intelligent User Interfaces. (2000) 106–112
10. Mobasher, B.: 3. In: Data Mining for Web Personalization. LNCS 4321 - Brusilovsky, P. and Kobsa, A. and Nejdl, W. (2007) 90–135
11. Lu, L., Dunham, M., Meng, Y.: Mining significant usage patterns from clickstream data. In: 7th International Workshop on Knowledge Discovery on the Web, (WebKDD'05). (2005)
12. Shyu, M., Haruechaiyasak, C., Chen, S., Zhao, N.: Collaborative filtering by mining association rules from user access sequences. In: International Workshop on Challenges in Web Information Retrieval and Integration. (2005)
13. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the International Conference on Data Engineering (ICDE'95). (1995) 3–14
14. Brin, S., Motwani, R., Ullman, J., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. (May 1997) 255–264
15. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: ACM Conference on Electronic Commerce. (2000)
16. Yong, W., Zhanhuai, L., Yang, Z.: Mining sequential association-rule for improving web document prediction. In: Proceedings of the Sixth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'05). (2005)
17. Kim, C., Kim, J.: A recommendation algorithm using multi-level association rules. In: Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03). (2003)
18. Wolff, R., Schuster, A.: Association rule mining in peer-to-peer systems. IEEE Transactions on Systems, Man and Cybernetics (2004)
19. Tan, X., Yao, M., Xu, M.: An effective technique for personalization recommendation based on access sequential patterns. In: Proceedings of the IEEE Asia-Pacific Conference on Services Computing (APSCC'06). (2006)
20. Deshpande, M., Karypis, G.: Selective markov models for predicting web-page accesses. In: First SIAM International Conference on Data Mining. (2001)
21. Xiao, Y., Dunham, M.: Efficient mining of traversal patterns. Data and Knwoledge Engineering **39**(2) (2001) 191–214
22. Huang, Y., Kuo, Y., Chen, J., Jeng, Y.: Np-miner: A real time recommendation algorithm by using web usage mining. Knowledge-Based Systems **19** (2006) 272–286